# DECISION SUPPORT SYSTEMS: LESSONS FOR THE 80's

Peter G. W. Keen

June 1981

CISR No. 70

Sloan WP No. 1223-81

Center for Information Systems Research
Massachusetts Institute of Technology
Alfred P. Sloan School of Management
50 Memorial Drive
Cambridge, Massachusetts, 02139
617 253-1000

# DECISION SUPPORT SYSTEMS: LESSONS FOR THE 80's

Peter G. W. Keen

June 1981

CISR No. 70

Sloan WP No. 1223-81

Center for Information Systems Research

Sloan School of Management

Massachusetts Institute of Technology

## 1. INTRODUCTION: Towards the Free Market

Five years ago the concept of Decision Support Systems was still a little hard to explain and relatively few managers saw DSS as much more than a fad. Now, they are well-established as an important, practical and intellectual contribution to the effective application of computers to organization planning and decision making.

The change reflects effective preaching and practice. It has also been greatly stimulated by the increasing move from computing being a monopoly to its becoming a free market. Throughout the 1970's, in most organizations, computers were effectively owned by a centralized Data Processing group. They were viewed as a cost to be charged out. The technology was relatively monolithic and users had little discretion to go to outside suppliers, even though the average backlog of committed projects was 2 years.

Since then, the term "computer technology" has become close to meaningless. The many technologies now range from micros to distributed systems, from general-purpose to end-user languages and application generators, and from stand-alone to in-house time-sharing to network utilities. The word "computer" does not do justice to telecommunications, which increasingly seems to be the strategic component and hardware the tactical one, or to voice input, facsimile, office systems, intelligent terminals, and network information services.

Every development in the computer-related technologies adds to variety of choice. It increases the differentiation of the building blocks for applications, stimulates the growth of new services and suppliers, and encourages users to shop widely and base their purchase on value rather than cost. Organizations may choose to regulate the free market and limit users' options in order to ensure integration, compatibility and coordination. Nonetheless, for most applications there is now more than one adequate technical solution, more than one supplier, and the "buy" option is often more attractive than the traditional "make".

Decision Support Systems were an explicit and early move towards the free market. From the start, DSS builders focused on the users' priorities. They offered a different style of computer system, linked more closely to key business activities than to the (vital) operational transactions that absorbed DP's attention and resources. They viewed the "quality" of a system from the users' rather than the technicians' perspective. They saw DSS as a service rather than a product and their own role as facilitators and consultants, not as programmers.

Data Processing was largely based on supply economics, hence its focus on cost, efficiency, charge-out (in contrast to incentive pricing), and centralized resource allocation. DSS reflect demand economics: service, fast delivery even where that means inefficiency, benefit rather than cost, and user control. Far too often, builders of information systems and models have worked from a naive model of the user. Decision Support has the sophisticated, responsive understanding of users and their work essential for the free market.

Ten years after Gorry and Scott Morton effectively first defined DSS, it seems most useful to review Decision Support in terms of what it has learnt about the demand side of computing. One strength of the DSS movement is its independence of any specific technology; technology is the means to the end of more effective management and decision making. Paradoxically, the increasing differentiation of the building blocks of the computer-related technologies seems to be pointing toward a convergence, rather than a divergence of systems. Data processing and word processing are merging (it is worth asking why they were ever considered separate). Electronic mail, enquiry systems, business graphics, micrographics and network information services will all be accessed from the same multifunction terminal. All the discretionary systems and services--those that individuals may choose to use in their own way (if at all), at their own pace, and for their own purposes--converge at the management work station.

This is where DSS began, at the terminal. DSS builders had first to learn how to market, and how to respond to discretionary users. The strategies it envolved are now of general relevance to any effort to exploit end-user languages, office systems, personal computers or information services. Building a DSS is the tactical issue. The strategic one is the effective application of any new means to the same overall end of improved managerial effectiveness.

## 2. Demand for Computer Systems

A monopoly obscures latent demand. If users cannot get delivery of systems, they do not bother to ask. Supply creates demand. The free market has transformed hidden, latent needs to overt--even strident--demand. It is becoming clear that the "official" backlog of two years was a gross underestimate; it may be closer to 10 years.

Several factors held back demand:

(1) shortage of personnel for development.

(2) the poor quality of what was available.

(3) the nature of development tools and techniques.

About 70% of the average DP group's personnel work goes to maintaining existing systems. Only 10 - 20% are available to develop new ones. Many companies are planning to expand their programming staff by 20 - 30% at a time when one in two jobs advertised goes unfilled. In this situation, someone has to assign priorities for development. Largely by default, DP has done so, since management has mainly played a passive role. The central mission of DP has been to automate the major operations of the organization, beginning with accounting and expanding to order entry, manufacturing, etc. It has been slow to respond to other needs, since to do so would divert resources at a time when there was a growing backlog of basic projects.

The main tools DP uses have evolved over a fifteen year period (beginning with the introduction of the IBM System/360). They were designed to facilitate the development of large and complex systems in which machine efficiency is important. They are not well-suited to ad hoc systems. The typical application system written in COBOL contains about 50 programs, each with 600 lines of code. Half the code is declarative and describes data files.

As we painfully and expensively learned in the 1960's and early 1970's, developing such a system is not at all easy. Program coding, the core of the process, amounts to only 10 - 20% of the effort. Testing can take 3 times as long. The systems development life cycle (SDLC) evolved through extensive trial-and-error learning as a methodology for planning and managing development. The SDLC emphasizes clear functional specifications. Together with the emerging methods of structured programming and structured analysis and design it has brought order to a haphazard process.

It has not, however, helped much in meshing supply and demand. The tools of DP work best with larger systems. They are not well-suited to meeting small, ad hoc needs. COBOL involves substantial set up costs and functional specifications take time to develop. The formal proceedures DP set up to ensure good management of complex projects impose delays. The backlog grows and small "one-off" projects get low priority. Managers are used to doing planning and analysis manually. They associate DP with transaction processing and standardized reports and see it as a distant, rather unresponsive unit.

Supply creates demand. Decentralized access to computers through outside time-sharing or stand alone desktops and minis, and special-purpose languages optimized for particular types of application and for fast delivery, have made ad hoc systems easy to build. End-user languages bypass the bottleneck imposed by lack of programmers and reduce dependence on DP. It is the combination of easy access to computer power and the availability of nonproceedural languages and related software that has created the free market. Obviously, they were made feasible only by the continued plunge in hardware costs. The tools of the free market are very inefficient in terms of machine usage.

Not only does a monopoly obscure the extent of potential demand, it totally hides the dynamics of consumer behavior. The cost-based management of large batch mainframes allowed users few trade-offs. Only when they had choices were their preferences made apparent.

Those preferences are based on value, rather than cost. Users are ready to pay a premium for ease of use and speed of delivery. Machine efficiency, the primary historical concern of DP, is of minor relevance to them. They have many, many needs for computer support. While they have been cautious in testing the water, once they dive in they frolic. In one company, for example, DP responded to growing pressures for service by allowing a free market for APL, a language noted for the remarkable speed with which systems can be built using it, and for its equally remarkable and more painful machine cost per hour. Within a year, expenditures on APL had grown to $20,000,000 of real money, spent by users out of their own budgets. A surprising number of these systems were for transaction-processing and reporting applications. Asked why they used APL for these, rather than COBOL or FORTRAN, users replied that they wanted the system quickly and that cost was a minor issue in comparison to time.

Delivery time and ease of use seem to be the key determinants of consumer preference in the free market. The DSS experience shows, too, that cost-displacement and cost/benefit analysis are of less relevance than "value-added" and qualitative benefits.

The list below derived from over 20 case studies indicates the main benefits identified by DSS users (See Keen, 1981):

(1) increase in the number of alternatives examined

(2) better understanding of the business

(3) fast response to unexpected situations

(4) ability to carry out ad hoc analysis

(5) new insights and learning

(6) improved communication

(7) improved management control

(8) cost savings

(9) better decisions

(10) more effective team work

(11) time savings

(12) making better use of data.

These are valuable benefits. They have not been provided by the DP monopoly. Two surveys, one in 1976 and one in 1979, showed that virtually all DSS are initiated and built outside DP, mainly by staff analysts in a functional area. The 1979 survey, by Wagner, also found that three-quarters of the 300 systems replaced manual proceedures. These DSS support strategic financial planning in some very large organizations, which have well-established and effective DP groups. It seems extraordinary in that context that planners and managers have bypassed the computer and vice versa. The barrier is not cost or ROI. The users of the financial planning language for DSS were uniformly pleased with the results, yet only 13% did any prior cost/benefit analysis and only 30% had any objective evidence of dollar benefits. They will pay for value and cost is secondary.

Alloway's analysis of the systems which managers have available in contrast to what they want indicates the overall volume of latent demand, much of which is for DSS. His study classified systems as Monitoring (standard reports), Exception, Inquiry and Analysis (mainly DSS). Sixty-three percent of the available systems are for monitoring and only 9% for analysis. Yet as the following table shows, of the "invisible" demand only 15% is for monitoring systems.

| | Monitor | Exception | Inquiry | Analysis |
|---|---|---|---|---|
| installed systems | 63% | 16% | 12% | 9% |
| under development | 36 | 22 | 28 | 14 |
| backlog | 23 | 9 | 36 | 32 |
| invisible demand | 15 | 12 | 38 | 35 |

## 2. Development Strategies for Meeting Demand

Decision Support taps this invisible demand. It mainly addresses its efforts to the area DP has generally been least interested in: Management tasks which are not particularly well defined, are small in scale and involve ad hoc needs, and "value-added" applications. More importantly, though, it has developed distinctive strategies for identifying and responding to users' needs. The latent demand for computer applications is a general opportunity - for DP, software vendors, consultants, office systems specialists, etc. The demand is not for traditional systems and it cannot be met through traditional techniques.

For example, managerial tasks are not routine. That makes "functional specs" hard to write. Users have shown they place a premium on speed of delivery: COBOL and the Systems Development Life Cycle are ill-suited to this. Managers' latent needs are for a type of support that provides the qualitative benefits that can only be summarized as "helping me do a better job". The cost-displacement mentality and lack of knowledge of how the user thinks and works found in too many DP groups lacks empathy, understanding, and credibility for this.

Decision Support had to evolve its own flexible, responsive techniques for developing this new style of system. As a result, we now have what amounts to a textbook on adaptive design. It defines:

(1) a user-oriented equivalent to systems analysis (rather than a data - or functionally - oriented one).

(2) an architecture for DSS, that emphasizes flexibility and ease of modification and extension.

(3) a phased development strategy, that relies on effective use of prototypes.

(4) a focus on the user-DSS dialog rather than data structures and algorithms.

(5) an R & D, rather than product focus, which reduces risk, and that facilitates and benefits from, user learning.

(6) a reliance on special-purpose languages and application generators that optimize speed of delivery rather than machine cost.

These techniques are all described in detail in many papers (Sprague 1981, Keen 1980). The following statement summarizes them:

> Building a DSS begins with the identification of a key decision or activity. Before trying to define the system, the designer looks carefully at how the job is currently done and at the user's vocabulary and mode of analysis. He or she then evaluates where the process could be improved, both from the users' viewpoint and from a more normative, conceptual perspective.
>
> As quickly as possible, the designer builds an initial system that is easy to learn and use. It may contain only a few functions. These will generally be ones that the users will find helpful in their current mode of operation. It may also include routines that substantially improve the range, quality and depth of analysis the user can carry out.
>
> The architecture of the system is defined in terms of three components:
>
> (1) the dialog manager: this is the flexible, largely self-explanatory software interface between the user and the functional routines contained in the DSS. The interface _is_ the system and its usability determines the usefulness of the DSS.
>
> (2) the data manager.
>
> (3) the functional routines.

The system is designed on a top-down basis with each of the components being independent and easy to change. The designer chooses a representation methodology: menu-driven, command-driven or a combination of both. The vocabulary, syntax and input and output formats are explicitly selected to reduce the amount the user has to learn and to minimize the translation from the problem statement to the DSS function, and vice versa.

The development language used is one that allows fast development and modification of the DSS. The initial system - "Version 0" - is built as quickly as possible, without asking the user to provide functional requirements and specifications. It represents the designer's interpretation of what meets the uses explicit and implicit needs. By directly working with it, the user can provide concrete reactions and help the designer evolve the full system. The designer users version 0 to learn about and from the user.

Since the benefits of the DSS will often be qualitative, no formal cost-benefit analysis is done at this initial stage. Instead, the designer and user jointly indentify areas of value, such time savings, increase in number of alternatives that can be assessed, or improved communication. Version 0 is a low cost prototype. The full system is evolved, with each new phase being an incremental cost. This adaptive, iterative design process is continued as needed, and is viewed as a joint effort, and as R & D. Whenever a new function is added to the DSS, the designer first sketches out the dialog and checks it with the user.

This strategy is less structured than the SDLC, but is not haphazard. It is an iterative cycling, by which a complex system is evolved out of simple components, at low-risk.

It is in no way an exaggeration to say that this is the general development strategy for applying the discretionary, terminal-based, innovative technologies of the 1980's. It is the distinctive contribution of Decision Support (obviously, though, it draws on techniques from computer science, Data Processing and software engineering. In particular, adaptive development relies on top-down design and stepwise refinement).

The textbook is not complete. Data management for DSS is still somewhat ad hoc, as is the testing process. DSS are extraordinarily hard to test. The system is not stable, but constantly evolving and it does not have fixed inputs and outputs. An imaginative user quickly tries new combinations, often ones the designer could not have anticipated.

The history of applications systems development is one of gradual structuring, of moving art to craft (and perhaps, later, to science). Requirements analysis, functional specifications, top-down design, structured programming and $C_1$ coverage testing are all contributions to software engineering. Decision Support provides a development strategy for those situations where structuring is inappropriate or infeasible. Innovation is always at least a little experimental, unpredictable and opportunistic. Software engineering is not suitable for R & D applications. Perhaps over the next decade we will uncover the structure of decision processes and office tasks so that we will have feasible decision engineering and literal office automation. In the meantime the textbook of Decision Support is the best tool we have for bringing the new technologies to new users.

## 4. Support Systems

"Support, not replace." This the philosophical, attitudinal core of Decision Support that most clearly positions it for the free market. It really presents a view of computers as a consumer product. It assumes that users must be able to adapt the technology to their own perceived needs and processes; while they in turn learn from it and adapt to it, they determine the pace, degree and function of change. They control the system, not vice versa.

It is impossible to support individuals without knowing what they do, how they think, what doing a "better" job means to them, and what they need to have to be willing to adopt a new tool. By contrast, the "replacement" approach mainly requires knowledge of the task or function. One may build an optimization model for, say, the capital investment decision with little concern about the user's processes and mode of analysis. Building a DSS, by contrast, requires explicit attention to the user. (That DSS may contain the same optimization algorithm).

Decision Support is a concept of productivity. The aim is to apply technology to make system-plus-manager more effective than either alone. The automation/replacement approach of traditional Data Processing and Management Science is also a concept of productivity. It is closer to the economist's ethos of efficiency and largely defines productivity from an input-output perspective. It reduces the individual's autonomy by imposing a sequence or structure.

Much of the new fashionable literature on productivity is surely likely to bring the idea into disrepute. Its message too often is: make them work harder, rationalize, impose new procedures, control. It is frequently rather insulting. Managers and professionals may assume it is a useful approach to the productivity "problem" of blue-collar and clerical workers. It is doubtful they would accept it as applicable to themselves. <u>They</u> work hard, are capable, and their skills, experience and judgement cannot be replaced through naive automation. Their work is, in any case, non-routine.

One almost universal rule of management seems to be that we are in favor of more control of our subordinates but more discretion for ourselves. We approve of the ethos of efficiency as our model of productivity for _them_. _We_, however, are different. Our superiors need to understand that our work cannot be automated, though our skills can be augmented by selective application of computer technology. Any such effort must recognize that we are the best judges of how to use the technology.

Decision Support began at the middle and senior management level - hence the D in DSS. The concept of Support, and the approach to productivity it implies, seems more general and key to the future of MIS, Management Science and Office Technology. Office Technology is now at a cross-roads. We may try to automate secretarial functions and structure managerial communication, in which case we will relive the truly awful early history of Data Processing. Instead we may support those functions and start by assuming that secretarial work is at best almost routine. If we do so, we will pay much more attention to what secretaries do, look at the world through their eyes, emphasize ease of use and flexibility in systems design, and aim at convincing these consumers of the personal value to them of the products we market to them.

The concept of support, whether it leads to Information, Decision, Policy, Negotiation, Research, Executive, Organizational, Personal, Financial, or Marketing Support Systems, is truly liberating. Too often, technicians are in an implicit conflict with their clients. They spend too much time insisting that their products _ought_ to be used. Very few DP professionals or management scientists are heros in their organization, valued for their service and viewed as indispensable. Those that are explicitly view their role as Support. They _start_ from the users' perspective, accept the validity of their concerns, priorities and constraints and subordinate technique to service. They assume that the tools, models, data and systems they offer can be useful. They recognize that their responsibility is to make them usable.

In the free market, the new consumers - managers, professionals, and secretaries - are revealing their preferences. Any survey of software packages, end-user languages, word processing and on-line enquiry systems, etc., etc., indicates that the leading products are those that combine flexiability, ease of learning and ease of use. An inflexible system is one where the user must adapt to the system not vice versa and where an application exists but was not anticipated. A system is hard to learn if the designer did not know or care who the users were and tried to impose his or her own concepts and techniques on them. It is hard to use if there was no understanding of the task, context and process. The DSS movement has gone well beyond the motherhood concept of user involvement; it is a user-driven technology.

Any skilled technician who adopts the philosophy and attitudes of Decision Support is positioned to apply any new technology and knows how to mesh it with users' needs. There are still far too many DP and OR/MIS specialists who think systems analysis is an abstract process of rationalization, and implementation the take-it-or-leave-it development of abstract tools. That approach worked poorly even under the monoloply's protection. It is simply a waste of time in the free market. Support, not replace.

## 5. Towards the Intergrated Work Station

There is still much new research and improvements in practice needed for DSS: model management techniques and intelligent interfaces, better and more powerful development languages, a theory of the psychological and cognitive impact of graphics, data extraction techniques, efficient multi-criteria models, etc. Decision Support is currently skimming the market - tackling the fairly obvious applications which are largely handled manually and where computer support is easy to provide and the payoff substantial.

This is a sensible strategy, but the long-term value of the DSS approach will surely be its contribution to the inevitable merging of all the application areas - Data Processing, Word Processing, Computer-based Message Systems, Management Science models, Enquiry Systems, and Personal Computing - through the multifunction work station. There are many, many software problems to be solved before this will occur; but already from being something far away in a back room, "the" computer has moved forward into the user's culture. The terminal is the computer, from the user's viewpoint. The workstation is the window to the various applications and services. The user is an active component in the system instead of a passive one. With old batch systems, the user filled out a form and the system did the rest. In an on-line office support environment, the system is only as good as the user. One choice people have with discretionary systems is simply not to use them.

We can expect more and more of managerial, professional and clerical work to be mediated by the computer. It will not be automated - this dogmatic statement is based as much on values rather than logic; perhaps it should read "It will be ethically wrong to try to automate such work. Instead, we will support the workers". This is the challenge and opportunity for Decision Support. We cannot afford to repeat the mistakes Data Processing and Management Science made in the 1960's as we apply new technologies to the office. This time we have to understand and respect the culture we are trying to change, have to build in flexibility and make sure people can comfortably live with the tools we install and must make sure that we provide meaningful benefits to the users.

There is no reason to fail this time. The free market has stimulated demand; consumers will buy suitable services. The demand in turn is stimulating supply; if a manager wants a financial planning language, a word processing system that is "user-friendly" or a color graphics package, he or she could find twenty acceptable options just with a few phone calls. The monopoly had to work with clumsy, expensive, monolithic tools. The Office Support work station will be built out of high quality, responsive components.

In this context, the attitudes and development strategy of the system builders will be critical. Viewed in terms of technical sophistication, DSS are unimpressive in general. Decision Support is not a revolution and is modest in its goals. Surely, however, no other field has been as successful in getting computers actually used and delivering meaningful benefits to people who have litte experience with them. The concept of Support and the techniques it implies are both a powerful refocussing of attention and effort and a feasible approach to implementation. We really do not have to argue about exactly what a DSS is or justify Decision Support as a new "field". What people in DSS do, what they know about users and how they apply any technology justification are enough.

Computing is now a free market; the equations of productivity - hardware costs falling, labor costs rising - mean organizations see long-term productivity gains being possible only through selective application of technology. The natural evolution of office technology is towards the management work station. The ideas, methods and attitudes of the DSS field have much to offer here. In the 1970's Decision Support was a radical, somewhat isolated movement that only gradually made a convincing case. In the 1980's it will be the mainstream.

We are at a stage now where we can consolidate what we know about supporting managers and building DSS. Five years ago, it was hard to make the systems work; DSS has its failures just like DP and OR/MIS, but has not had them written up and dissected them so openly. Now, the technical tools on which DSS is based have improved so dramatically that it is rare for software and hardware to be a stumbling block. Decision Support is an application concept. The better and broader the technology available, the better and broader the support we will provide. The greater the forces of demand, the more we can supply. That is a healthy state of affairs for the second decade of Decision Support.

## REFERENCES

(1) Alloway, Robert M., "User Managers' Systems Needs", CISR (Center For Information Systems Research), Working Paper No. 56, May 1980.

(2) Gorry, G.A., and M.S. Scott Morton, "A Framework for Management Information Systems", Sloan Management Review, Vol. 13, No. 1, pp. 55-70, Fall 1971.

(3) Keen, P.G.W. and Scott Morton, M.S., Decision Support Systems, An Organizational Perspective, Addison-Wesley, Reading, MA, 1978.

(4) Keen, P.G.W., "Value Analysis: Justifying Decision Support Systems", September 1980.

(5) Sprague, R.H., "A Framework for Decision Support Systems", MIS Quarterly, Vo. 4, No. 4, 1980.

(6) Wagner, G.R., "Realizing DSS Benefits with the IFPS Planning Language", paper presented at the Hawaii International Conference on System Sciences, Honolulu, HI, January 1980.